元对象机制驱动的复杂系统开放式顶层建模

王西超[1]，曹云峰[1]，丁 萌[1]，庄丽葵[1]，王 彪[1]，杨 斌[2]

(1. 南京航空航天大学自动化学院 南京 210016; 2. 南华大学电气工程学院 湖南 衡阳 421001)

【摘要】为了在复杂系统开发过程的不同阶段，以及在不同建模工具和设计部门之间更加准确地描述系统的顶层模型，提出了一种新的面向复杂系统开放式顶层建模方法。基于元对象机制的元建模框架，对SysML进行基本衍型扩展和Simulink衍型扩展；在顶层建模中，把复杂系统划分为静态结构模型、动态行为模型以及两种模型间的交互关系；并对各抽象层次模型进行形式化定义，实现多种异构模型的有效集成。最后，在某无人机飞控系统顶层建模实例中，对基于顶层模型进行了飞控系统的统一建模描述，实现了复杂系统中多种异构模型的高层次抽象，验证了该方法对复杂系统顶层建模的有效性。

关 键 词 复杂系统； 元对象机制； 模型集成； SysML扩展； 顶层建模

中图分类号 TM391 **文献标识码** A doi:10.3969/j.issn.1001-0548.2012.04.001

Complex System Openended Top-Level Modeling Drivened by Meta Object Facility

WANG Xi-chao[1], CAO Yun-feng[1], DING Meng[1], ZHUANG Li-kui[1], WANG Biao[1], and YANG Bin[2]

(1. College of Automation Engineering, Nanjing University of Aeronautics & Astronautics Nanjing 210016;

2. College of Electrical Engineering, University of South China Hengyang Hunan 421001)

Abstract In order to describe top-level model in different development stages, tools and departments for the development of complex system more accurately, a novel openended top-level modeling method for complex system is proposed. Based on the facility of Meta modeling framework, SysML is profiled by basic stereotype and Simulink stereotype. In top-level modeling, the complex system is divided into static structure model, dynamic behavior model and the interaction between them. The formal definition of each abstract model is discussed and the integration of various structure models is realized. Take a flight control system of unmanned aerial vehicle as an example, the system is modeled uniformly based on top-level model and the higher-level abstraction of those isomerous models in complex system is implemented. The experimental results validate the effectiveness of the proposed modeling method.

Key words complex system; meta object facility; model integration; SysML profile; top-level modeling

系统根据组成子系统以及子系统种类的多少和它们之间的关联复杂程度，分为简单系统、简单巨系统和复杂巨系统。其中，子系统种类很多，又有层次结构，且关联关系很复杂的，称为开放的复杂巨系统(简称复杂系统)[1]。

随着计算机技术的发展，复杂系统建模与仿真技术已成为研究热点。传统的复杂系统建模方法为：首先基于各专业领域的CAx/DFx工具进行分系统建模，进而基于过程集成方法建立系统分析过程和设计修改过程，最终确定系统的结构、行为与关键数据。传统方法的缺点是自动化差，迭代过程复杂，不能完整地构造系统顶层的行为和结构，难以保证系统的整体性。另外，由于缺乏顶层建模方法，分系统之间各自独立建模，导致分系统设计之间的信息孤岛，设计人员之间难以协同工作。为适应复杂系统多学科建模的需要，不同学者提出了不同的建模方法。

1) 物理系统的结构行为直接建模。

该方法主要是基于图形化的符号表示系统的各个结构以及运行的动态特性，如 Simulink、Easy5 等。其优点是比较直观，系统模型由若干方块通过线段连接而成，每个方块对应一定的输入输出关系。但

收稿日期：2012－05－15
基金项目：国家自然科学基金(61102108)；航空基金(20110752005).
作者简介：王西超(1982－)，男，博士生，主要从事复杂系统建模与仿真、飞控系统设计与仿真方面的研究.

该方法的系统连接过多且方块间易误连，很难体现系统的分系统特征。

2) 基于统一机理的系统建模。

典型的有基于微分代数方程的物理系统建模语言Modelica、离散事件系统仿真(discrete event system simulation, DEVS)[2-3]等。Modelica支持多领域(如电子、机械、控制等)模型以统一的形式化方法表示，实现多领域协同建模[4-5]。但该类方法对系统机理界限明显，难以实现不同机理的(如连续、离散；定性、定量等)复杂系统建模。

3) 基于工具软件接口集成的系统建模。

为实现复杂系统跨领域的协同建模，相关学者研究了不同商业软件间的接口，通过接口集成各种软件。如文献[6]将Adams和Simulink通过接口实现了电动车机电混合控制建模；文献[7]则基于AMESim和Simulink的接口实现了混合动力车的机械、液压、控制系统的协同建模。该类方法能够充分利用现有商业软件，但在系统层的总体行为建模方面存在不足。

4) 基于组件思想的系统建模。

该类方法把仿真模型封装成为可以独立开发、运行的组件模块，通过一定范畴的接口描述，使模型的调度、调试、运行不受组件之间相互依赖的影响，使模型的集成可以自动进行，如DEVS、HLA[8-9]等。该类方法适应于实际系统的开发，但其缺少组件间形式化的多学科集成机制，如在HLA标准中难以实现联邦的形式化顶层建模。

综上所述，当前的复杂系统建模技术在多学科异构模型的有效集成，以及系统层的建模上还存在不足。因此，需要一种顶层建模技术能够从系统层描述复杂系统的行为和结构，进而实现复杂系统分析和设计过程中不同建模工具、不同设计部门之间的协同建模。本文提出了一种有效的顶层建模方法，即通过引入元对象机制高度抽象多学科异构模型，将其统一为由元模型实例化而来的仿真组件模型，从系统层对各种异构仿真模型进行一体化建模描述，实现复杂系统顶层建模。

1　顶层建模知识

1.1 顶层建模技术

顶层建模技术是一种基于不同领域模型搭建、描述复杂系统的建模方法。所谓顶层建模是在系统层次上对系统静态结构、动态行为的描述。静态结构部分描述了各层次、各学科模型的功能模块分解、接口和连接关系等；动态行为模型则描述了系统动态过程以及对静态模型的调度。

1.2　元对象机制

对象管理组织(object management group, OMG)的元对象机制(meta object racility, MOF)是模型驱动体系结构(model driven aichitecture, MDA)中关于如何抽象建立模型对象、模型的语义或模型之间如何集成和互操作等信息的描述规范[10]。MOF定义了元模型的基本元素、语法和结构，定义了模型的4个元层次：元元模型、元模型、模型和实例[10]。

复杂系统顶层建模采用MDA和MOF的元模型分层思想，将顶层建模框架定义为一个4层次的模型体系架构，如表1所示。元元模型经实例化为元模型，元模型进一步实例化为原子模型和耦合模型。通过采用MOF层次化的模型结构，可以实现具体应用与模型抽象间的分离，从而保证了复杂系统中的各种多学科异构模型在模型抽象上的统一。

表1　MOF与复杂系统顶层模型的4层次结构

MOF元层次	模型描述	SysML	复杂系统顶层模型
元元模型(M_0)	最高抽象层次，定义最基本的类、关联等元素	数据结构、属性	数据结构、属性，状态
元模型(M_1)	元元模型的实例，定义模型描述语言的基本要素	类、对象	扩展类
模型(M_2)	元模型的实例，定义描述某应用领域的顶层模型	“driver”类、“car”类等	原子模型
实例(M_3)	模型的实例，定义具体领域的数据值	驾驶员Jack、宝马汽车等	耦合模型

2　顶层建模层次化规范

顶层建模中系统划分为仿真对象模型(simulation object model，SOM)和仿真联邦模型(simulation federation model，SFM)。SOM由原子模型(atomic model，AM)、耦合模型(coupled model，CM)、模型之间的连接和行为模型(behavior modle，BM)组成。AM、CM和BM是顶层系统建模的主要模型元素，顶层模型数据交换格式采用可扩展标记语言XML，实现顶层模型在不同工具软件之间的模型传递。系统层次化结构如图1所示。

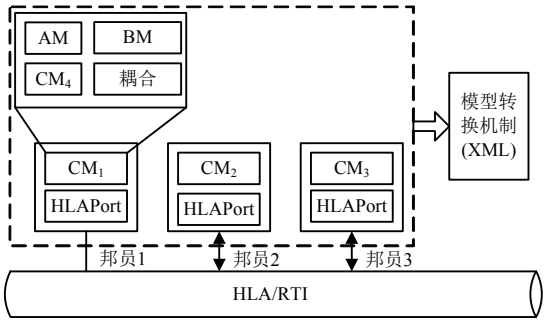

图1　顶层建模的系统模型层次

2.1 原子模型形式化

AM是顶层建模中的基本模型元素，它从功能和结构上是不可再分的，具有高度的重用性。AM与外部模型的信息传递主要通过端口进行(包括输入/输出数据端口、输入/输出事件端口)。AM模型结构如图2所示。

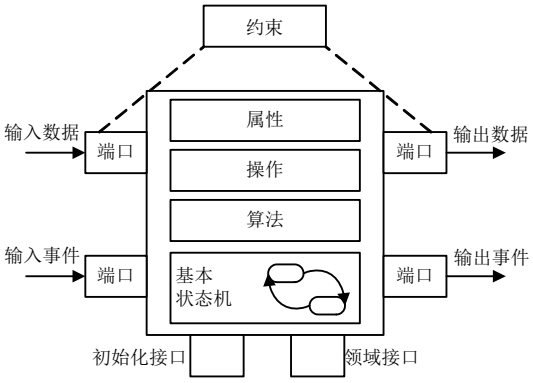

图2 AM模型结构

AM的形式化描述如下：

AM：<输入数据，输出数据，输入事件，输出事件，约束，初始化接口，领域接口，属性，操作，算法，基本状态机>；

输入数据={ip_1, ip_2,…, ip_n}，表示输入数据端口集合；

输出数据={od_1, od_2,…, od_n}，表示输出数据端口集合；

输入事件={ie_1, ie_2,…, ie_n}，表示输入事件端口集合；

输出事件={oe_1, oe_2,…, oe_n}，表示输出事件端口集合；

约束：表示输入输出数据端口之间的约束关系，一般情况下为基本数学表达式，如$f(od_1, od_2,\cdots, od_n) \leqslant 0$；

初始化接口：表示对端口进行初始化，包括输入/输出数据端口和输入/输出事件端口；

领域接口：表示模型附言，主要表示模型所属学科领域；

属性：同SysML里类的atrribute；

操作：同SysML里类的opertation；

算法：表示模型仿真算法，包括基本函数算法和复杂算法；

基本状态机：模型的基本行为状态机，通过对仿真过程的控制来描述系统动态行为；

基本状态机=<S, S_0, ie_0, g>

S= {(“INIT”, A_1), (“RUN”, A_2), (“STOP”, A_3), (“PAUSE”, A_4), (“EXIT”, A_5)}

A_i是S_i的动作，表示对仿真算法的调度；

ie_0= {evInit, evRun, evStop, evPause, evTimeAdvance};

g：状态转换集合，$S\times ie_0 \rightarrow S$。

2.2 耦合模型形式化

复杂系统通常由大量多学科异构模型多层次复合而成。因此，在AM的基础上，加入AM端口间的连接来描述CM，其形式化描述如下：

CM：<{接口}，{AM, CM}，{耦合}>。

其中，接口表示模型输入输出/数据端口和输入/输出事件端口；AM定义如2.1节所述；耦合= < {关联}，{约束}>，表示CM的耦合包含关联和约束两种形式。CM本身是自嵌套的模型形式，其内部组成除了可以是不可再分的AM，也可以是其他的CM，如图3所示。

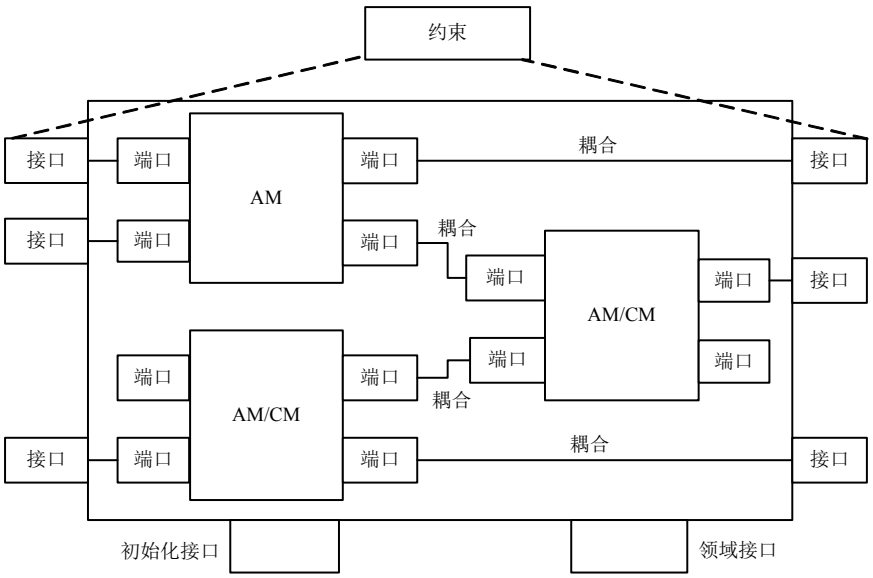

图3 CM模型结构

2.3　行为模型形式化

复杂系统的行为模型BM是在时间域内描述交互信息的发生过程，描述各静态模型对各交互信息的反应，通过对各个静态模型的层次化调度实现复杂系统的动态行为。顶层建模采用状态图(或者活动图/顺序图)反映时间域交互信息的发生过程和对象系统的行为。状态图关注一个对象的生命周期内的状态及状态变迁，以及引起状态变迁的事件和对象在状态中的动作等。主要刻画和描述一个对象的生命周期内的状态改变和对外界事件和时间的响应能力。其形式化描述如下：

BM：< inputEvent, *t*, guard, *S*, action, outputEvent >。

inputEvent={ie_1, ie_2,…, ie_n}，特指状态的外部触发事件集合；

t：逻辑时间判断，一旦达到设定时间，状态发生迁移；

guard：条件判断，当模型满足特定条件时，状态发生迁移；

S={S_1, S_2,…, S_n}，表示模型状态集合；

action：表示模型动作；

outputEvent={oe_1, oe_2,…, oe_n}，特指状态输出事件集合。

2.4　组件模型

组件(或成员)是支持HLA标准的仿真联邦成员。如图1所示，在复杂系统协同仿真中，组件特指基于顶层建模集成开发环境开发的符合HLA标准的联邦成员，即：

组件= HLAPort+ AM/CM+BM

顶层建模方法通过严格的形式化方法，描述复杂系统的行为和结构，通过建立基于原子模型—耦合模型—组件(成员)模型—联邦模型的静态结构框架，以及基于状态、事件和时间的动态行为模型，描述系统在一定的时间域内，如何通过模型之间的层次调度和事件传递完成复杂系统的顶层建模。

3　顶层模型的实现

前面进行了顶层模型的形式化定义，基于MOF，结合复杂系统顶层建模需求，对SysML进行扩展实现顶层模型。

SysML[11-12]是OMG及国际系统工程学会为满足系统工程领域建模需要，基于MOF，在重用UML2.0[13]子集基础上经过扩展而推出的面向对象图形化建模语言。虽然SysML能够支持各种系统的需求分析、系统设计、验证和确认等，但SysML在复杂系统顶层建模方面尚存在不足，需对其进行扩展。SysMLprofile是SysML标准提供的扩展机制，SysMLprofile由特定的衍型(stereotype)、约束(constraints)和标记(tagged values)组成。

3.1　SysML静态模型扩展

在复杂系统的顶层静态结构建模中一个重要的定义是定义各种层次、耦合关系；而在SysML中却不能直接支持模型的耦合关系，原因是SysML关注的是类之间的交互而非接口关系。本文的思路是在SysML的基础上，为类加上事件端口(standardport)和数据端口(flowport)，从而明确类之间的交互是通过类的输入/输出进行的；本文通过SysMLstereotype提供对该类层次、耦合的扩展。

1) 原子类：是对SysML中的类进行扩展后的一种stereotype。原子类是在原类的基础上，为类增加端口(包括standardport和flowport)；原子类之间的交互通过端口进行。

2) 耦合类：把关系紧密的原子类通过一定形式的连接关系集合在一起，形成耦合类，共同完成一项任务。

3.2　SysML动态模型扩展

在复杂系统中，为能更好地理解模型的动态特性，定义两种基本的模型状态转移：内部转移和外部转移；在SysML的状态图中没有提供这种支持，本文通过stereotype支持两种转移特性。

1) 内部状态stereotype(<<Inside>>)：通过stereotype提供对状态转移的扩展，表示不需要外部事件的激励在本状态的生命周期内的状态转移。

2) 外部状态stereotype (<<Exterior>>)：通过stereotype提供对状态转移的扩展，表示当外部某一事件发生时，从本状态将跳到别的状态。

3.3　SysML的Simulink模型扩展

复杂系统动态行为包括离散行为和连续行为，但SysML对连续行为的建模能力不足，鉴于Simulink是目前连续行为建模领域内较为通用的语言，本文通过SimulinkStereotype实现SysML连续行为的扩展。

由于SysML模型与Simulink模型是两种性质的异构模型，在进行SysML的Simulink模型扩展时需解决模型的调用方式、模型的数据接口和数据类型识别3个问题。表2给出了SimulinkBlock的stereotype所

包含的tagged values详细定义。

表2 SimulinkBlock的stereotype定义

扩展类型	成员名称
tags	IncludePath
	MatlabRoot
	SimulinkProjectFile
	SimulinkSampleTime
	SimulinkCodeDir
	SimulinkSourceFiles

3.4 顶层模型的实现

经SysML扩展后的原子模型如图4所示。扩展以类为元模型，如“输入事件”和“输出事件”通过在类上扩展“标准端口”实现；“输入数据”和“输出数据”通过在类上扩展“数据流端口”实现；“基本状态机”通过在类上扩展Statechart实现；“算法”则是通过SimulinkStereotype扩展实现。其中“ ”表示类动态模型扩展，<<Inside>>和<<Exterior>>分别表示对状态图的内部状态扩展和外部状态扩展；“ ”表示类的Simulink模型扩展。

可以看出，原子模型对应于SysML扩展的原子类、状态图和SimulinkStereotype；耦合模型对应于SysML扩展的耦合类、状态图和SimulinkStereotype；行为模型对应于SysML的状态图扩展。

4 顶层建模集成开发环境

为支持复杂系统顶层建模，本文设计了顶层建模集成开发环境，由前面分析可知，顶层模型包括AM、CM、BM和FM，其中AM、CM、BM均由SysML扩展而来。本文选用Rhapsody软件作为SysML建模工具，并以其为核心，集成Matlab/Simulink、Doors、VC++和Oracle共4个商用软件搭建顶层建模集成开发环境，整体结构图如图5所示。其中Doors平台用于对复杂系统的用户需求进行管理；Rhapsody平台利用SysML需求图和用例图进行复杂系统需求分析，分别利用块图/对象图和状态图/顺序图设计静、动态离散模型；Matlab/Simulink平台用于进行Rhapsody的SimulinkStereotype扩展，实现原子模型中的复杂算法建模；利用VC++开发测控界面连接Rhapsody、Matlab/Simulink和Oracle；VC++采用ADO数据库访问技术，与Oracle数据库建立关联，以实现对复杂系统开发过程中的数据、文档和模型的存取，其中Doors平台中的用户需求能够和Rhapsody平台中的需求模型双向动态关联；为支持HLA联邦整体设计，开发了Rhapsody与HLA的HLAPort以及Matlab与HLA的HLAPort。该平台能够以全数字化的方式完成复杂系统的需求管理、需求跟踪、系统分析及顶层建模仿真。

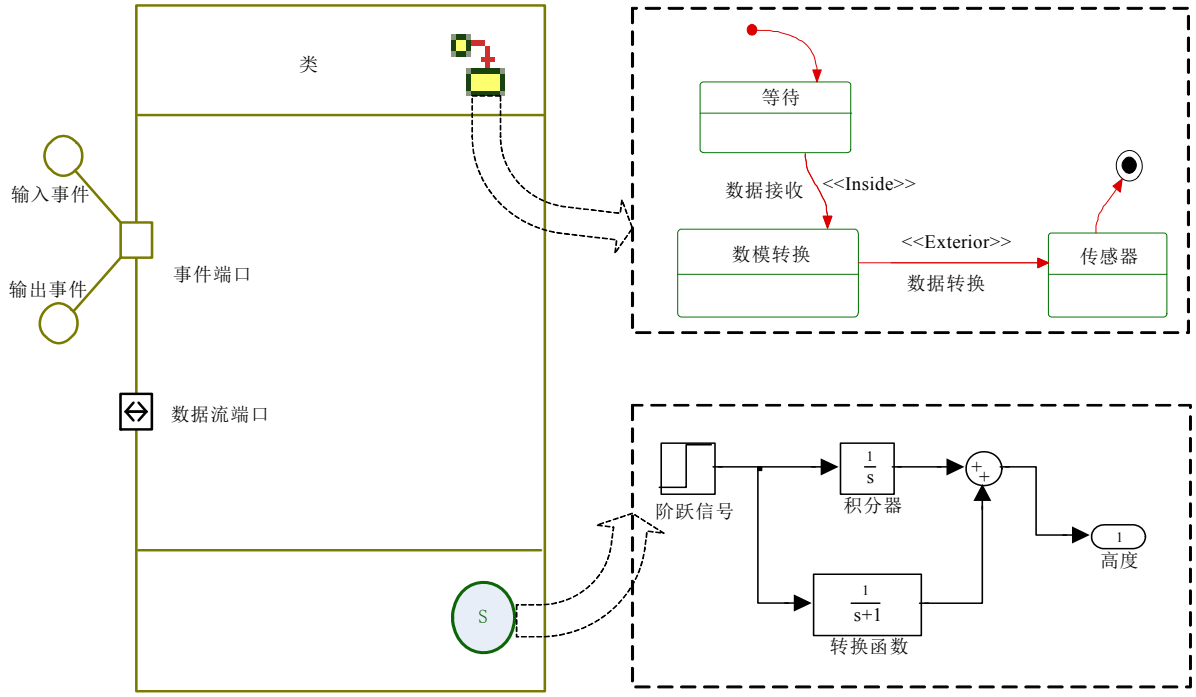

图4 基于SysML扩展的原子模型

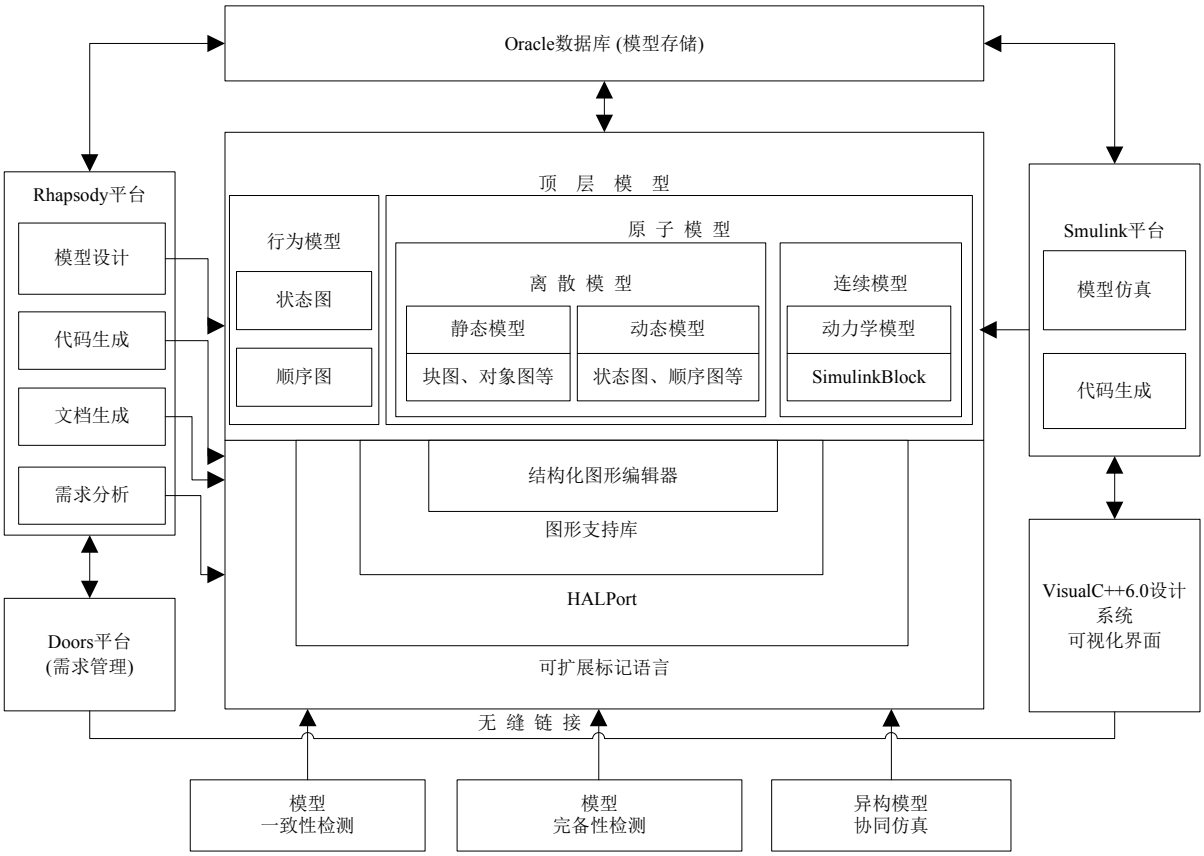

图5　复杂系统顶层建模集成开发环境

5　设计实例

5.1　无人机的飞控系统顶层建模

本文以某无人机(unmanned aerial vehicle, UAV)的飞控系统(flight control system，FCS)这一典型复杂系统作为顶层建模的应用实例。FCS顶层模型主要由UAV多体动力学模型、发动机动力传动系统模型、舵机电子系统模型和飞控计算机控制模型4部分构成。其中，UAV多体动力学模型接收动力传动系统模型传来的发动机转速，以及电子系统模型传来的作用力和作用力矩，通过解算输出UAV的姿态角速度、姿态角、速度和位置；动力传动系统接收控制系统给出的风门大小信号和多体动力学系统反馈回来的速度信号，通过解算输出发动机推力；电子系统模型接收控制系统给出的脉冲开关信号和多体动力学系统反馈回的姿态角速度信号，通过解算输出作用力和作用力矩；控制系统模型接受外部的地面站控制指令开始动作，在解算过程中不断接收UAV多体动力学模型反馈回的4组信号，输出脉冲开关信号控制电子系统和动力传动系统。

基于扩展SysML和顶层建模集成开发环境的图形化界面对FCS进行顶层建模，上述4部分模型被统一描述为原子模型AM(如图6中黑色小矩形框)，这些AM间通过Interface(如图6中方形端口表示事件端口，内嵌双向箭头的方形端口表示数据流端口)进行耦合连接。需要说明的是，这些原子模型AM实际上是一种抽象模型，而非基于领域软件建立的实体模型(如Simulink建模的控制律模型等)。FCS顶层模型则为由这4个原子模型AM通过耦合连接构成的耦合模型CM，如图6中下方大矩形框。可以看出，CM尽管在内部还包含具体的子模型和复杂的映射连接关系，但在外部表象上，FCS顶层模型CM仅暴露了特定的Interface与外界交互。因此，FCS顶层模型CM在更高层次的应用中，如FCS优化系统，又可以作为原子模型AM进行重用，与多学科优化模型(如图6上方大矩形框)耦合连接，构成更大规模的FCS优化系统模型。

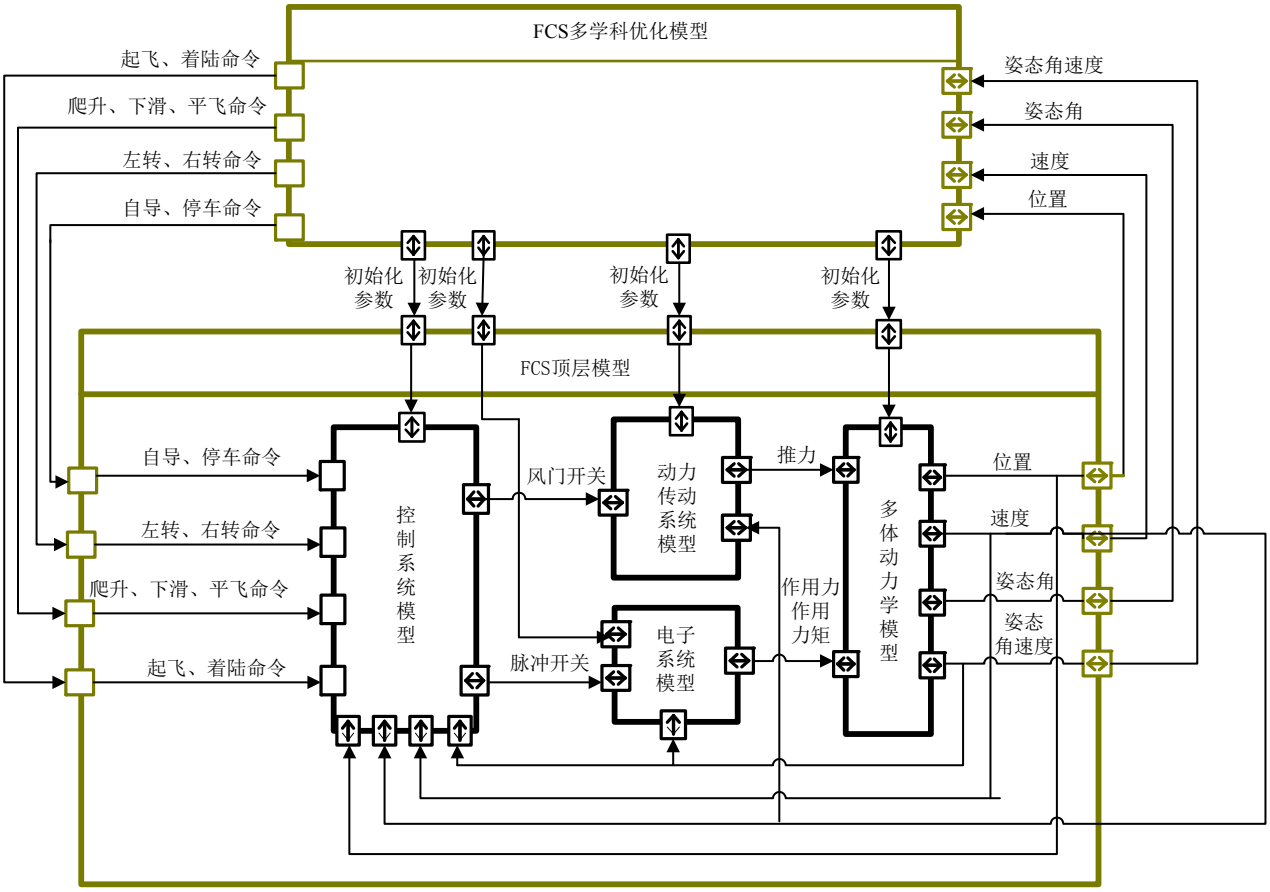

图6 飞控系统顶层模型静态结构建模

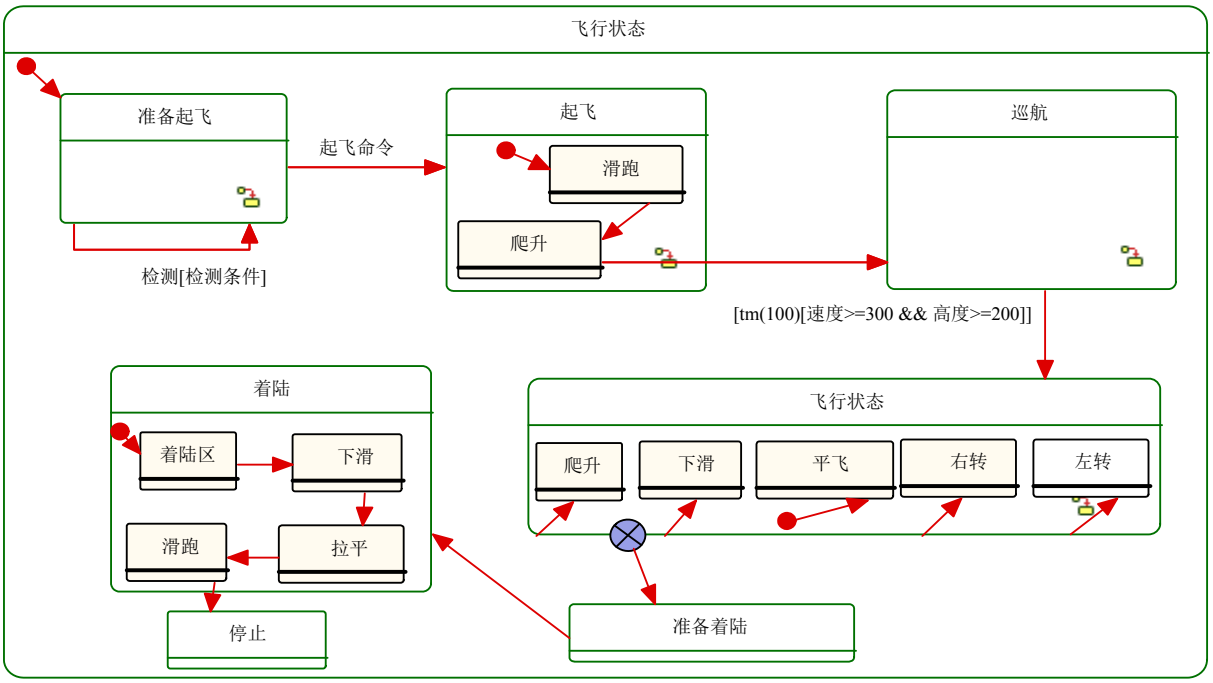

图7 飞控系统顶层模型动态行为建模

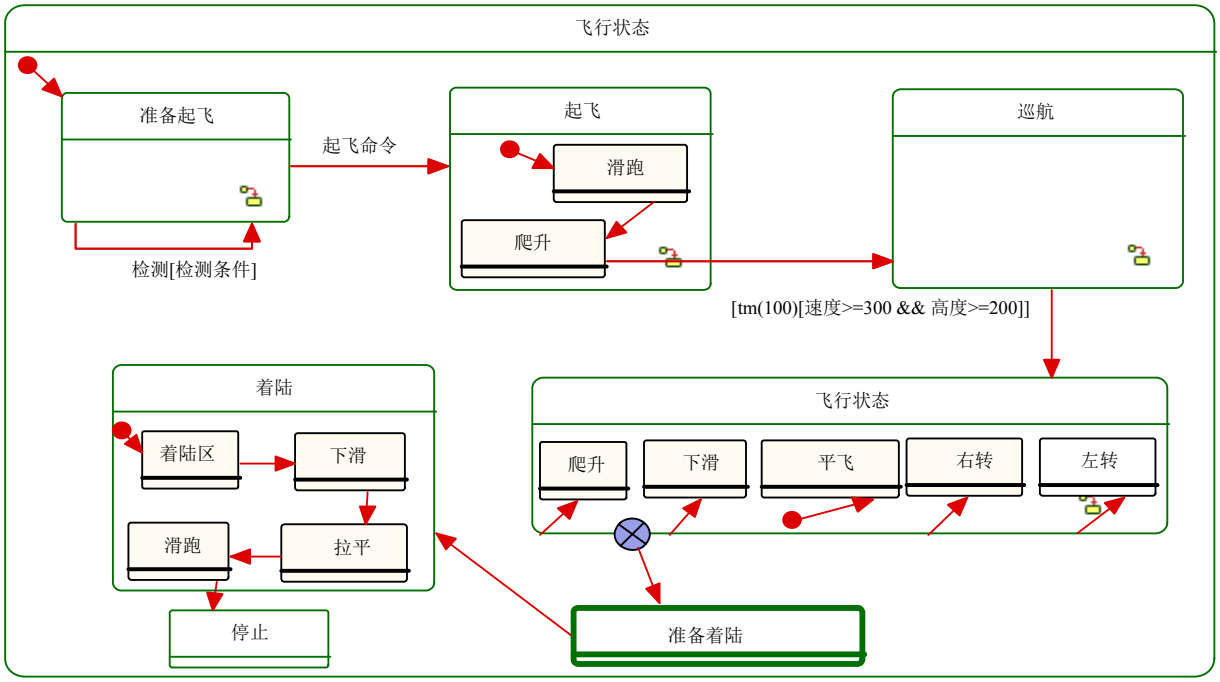

图8　图7的某时刻仿真图

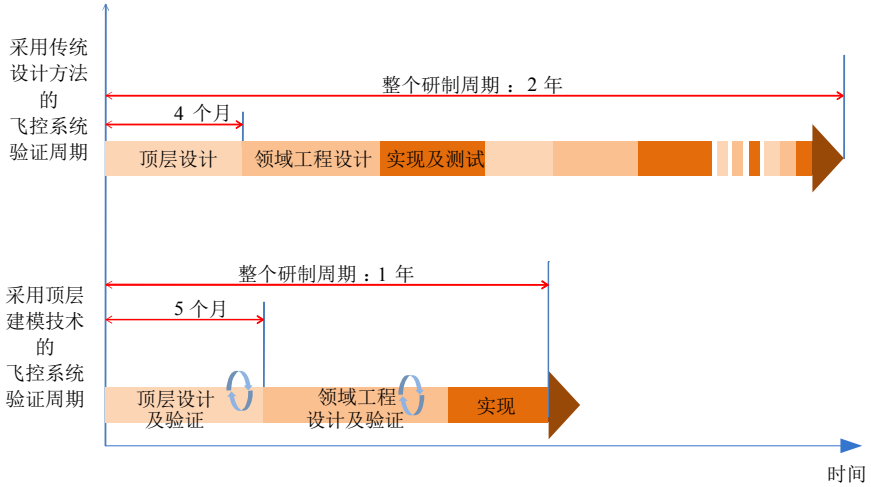

图9　飞控系统研制周期对比

在领域建模时，UAV多体动力学模型、舵机电子系统模型都是连续系统模型，分别用Adams和Simulink软件构建；发动机动力传动系统模型是连续和离散时间混合的系统模型，用Easy5软件构建；控制系统模型是一个连续和离散事件混合的系统模型，其内部映射通过状态图进行描述，使用Rhapsody软件进行建模。对于复杂系统顶层建模人员而言，它们都是黑盒模型，即模型内部是封闭的，在实现时通过在每个周期调用相应软件求解器向前解算指定的步长，并获取所需的输出。

FCS顶层模型的静态结构模型如图6所示，动态行为模型如图7所示，图8为图7在某时刻的仿真结果图。由FCS顶层建模过程可以看出：

1) 顶层建模在抽象模型阶段(即原子模型AM和耦合模型CM)就可对复杂系统进行静态结构和动态逻辑行为的建模及仿真验证，对各种设计方案及设计决策进行检查和评估。

2) 顶层模型清晰的静态结构和动态行为逻辑可以指导领域建模，避免领域建模时由于多领域模型的逻辑混乱而导致多次迭代循环设计，提高了复杂系统设计效率。

3) 在进行复杂系统顶层建模时，只需对复杂系统中异构原子模型AM的接口进行一定形式的连接，便可将这些异构原子模型AM统一建模为耦合模型

CM形式，实现原子模型级的高度模型重用。

4) 领域模型(如多体动力学Adams模型)可以封装为原子模型形式，与顶层的原子模型/耦合模型进行混合仿真，验证领域模型功能。

5.2 应用效果

根据用户单位复杂系统顶层建模技术前后的无人机飞控系统研制情况对比如图9所示。从图中可以看出，使用传统的设计方法，飞控系统研制需要在顶层与领域工程层次之间经过多次反复的迭代，研制周期长达两年时间。采用顶层建模技术后，虽然顶层设计周期比过去稍长，但由于显著提高了设计质量，因此不会在顶层与领域工程层次之间出现反复，在同等的人力投入下研制周期缩短为过去的一半，研制经费降低约1/4。

6 结 论

本文以复杂系统多学科设计工程应用为背景，以解决多学科异构模型的统一建模为目的，基于元对象机制，对SysML进行扩展，研究了一种复杂系统顶层建模方法，该方法将复杂系统顶层建模分为静态结构建模和动态行为建模，从而使系统设计人员可以在复杂系统开发早期对各种设计方案及设计决策进行检查和评估。该建模方法有以下特点：

1) 顶层建模与模型实现组件相分离。通过将多学科异构模型抽象为原子模型、耦合模型和行为模型，顶层建模仅描述了模型组件的接口和组件间的耦合关系，并没有描述组件内部的实现细节，从模型抽象层次上将各种异构模型间的信息交互与模型实现相分离，从而减弱模型间的差异性，能够描述由多种类型(连续、离散和混合)仿真模型组成的复杂系统。

2) 采用模型组件封装多学科模型。多学科模型被封装为原子模型组件或耦合模型组件，模型组件只关注其对外的接口和行为，仅描述其领域特征，因此提高了多学科和多层次模型的可组合性以及模型的可重用性。

3) 实现复杂系统的多层次建模。从原子模型、耦合模型到成员模型，模型抽象度与分辨率逐次变低，既支持高层次的模型抽象和自顶向下的系统分析，也支持通过逐步复合模型的自底向上的系统设计与构造。

参 考 文 献

[1] 钱学森, 于景元, 戴汝为. 一个科学新领域——开放的复杂巨系统及其方法论[J]. 自然杂志, 1990, 13(1): 3-10.
QIAN Xue-shen, YU Jing-yuan, DAI Ru-wei. A new discipline of science——The study of open complex giant system and its methodology[J]. Ziran, 1990, 13(1): 3-10.

[2] WAINER G, LIU Q. Tools for graphical specification and visualization of DEVS models[J]. Simulation, 2009, 85 (3): 131-158.

[3] WAINER G, GLINASKY E, GUTIERREZ A M. Studying performance of DEVS modeling and simulation environments using the DEVS tonebenchmark[J]. Simulation, 2011, 87(7): 555-580.

[4] IMSLAND L, KITTILSEN P, SCHEI T S. Model-based optimizing control and estimation using Modelica models[J]. Modeling, Identification and Control, 2010, 31(3): 107-121.

[5] HEDIN G, AKESSEN J, EKMAN T. Extending languages by leveraging compilers: From modelica to optimica[J]. IEEE Software, 2011, 28(3): 68-74.

[6] SHU Hong-yu, XU Yong-liang, CHEN Qi-ping, et al. Simulation on driving system used for differential steering of electric scooter[J]. Transactions of Tianjin University, 2011, 2(17): 103-106.

[7] SONG Chuan-xue, WANG Ji, JIN Li-qiang. Study on the composite ABS control of vehicles with four electric wheels[J]. Journal of Computers, 2011, 3(6): 618-626.

[8] OUNNAR F, PUJO P, MEKAOUCHE L, et al. Integration of a flat holonic form in an HLA environment[J]. Journal of Intelligent Manufacturing, 2009, 1(20): 91-111.

[9] ADAK M, TOPCU O, OGUZTUZUN H. Model-based code generation for HLA federates[J]. Software-Practice and Experience, 2010, 2(40): 149-175.

[10] OMG. MOF, Version1.4 [EB/OL]. [2010-01-05]. http:// www. omg. org/ technology/ documents/formal/mof.htm.

[11] OMG. OMG SysML specification v1.1. [EB/OL]. [2010-05-06].http://www.sysmlforum.com/docs/specs/OMGSysML-v1.1-08-11-01.pdf. 2008.11.01/2010.05.10.

[12] CONSTANTINE J A, SOLAK S. SysML modeling of off-the-shelf-option acquisition for risk mitigation in military programs[J]. Systems Engineering, 2010, 13(1): 80-94.

[13] OMG. UML2.2[EB/OL]. [2010-01-05]. http:// www.omg. org/spec/UML/2.2/ Infrastructure.

编 辑 蒋 晓